Tracking under Uncertainty with Cooperating Objects using MOMDPs

Jesús Capitán[1], Luis Merino[2], and Aníbal Ollero[1,3]

[1] University of Seville
Camino de los Descubrimientos s/n
41092, Spain
jescap,aollero@cartuja.us.es
[2] Pablo de Olavide University
Seville, Spain
lmercab@upo.es
[3] Center for Advanced Aerospace Technology
C. Wilbur y Orville Wright 17-19-21
41309, La Rinconada (Spain)

Abstract. This paper presents how *Partially Observable Markov Decision Processes* (POMDPs) can be used for Cooperating Objects control under uncertainty. POMDPs provide a sound mathematical framework to deal with planning actions when tasks outcomes and perception are uncertain, although their computational complexity have precluded their use for complex problems. However, considering *mixed observability* can lead to simpler representations of the problem. The basic idea is to assume that some of the components in the state are fully observable, which is reasonable in many applications. In this paper, target tracking by means of a team of mobile Cooperating Objects (for instance, robots) is addressed. Instead of solving an usual POMDP, the problem is solved under the assumption of mixed observability, which will reduce the complexity of the original model. Moreover, a decentralized solution in which the COs have no knowledge about others' actions is proposed. In this sense, a implicit coordination will be derived from the sharing information among the COs. Finally, some simulations are presented to show the advantages of the methods proposed and how the mentioned coordination arises by using heterogeneous COs with different behaviors.

1 Introduction

Most (if not all) applications of Cooperating Objects (COs) require to deal with uncertainty. Regarding sensing, this requires to be able to maintain an estimation of the quantities of interest and their uncertainties, by means of what is usually called a belief. Moreover, when planning and control of COs are considered, the objective is to determine the set of actions that are optimal in some sense, taking into account all the sources of uncertainty.

In the literature, techniques for decision-theoretic planning under uncertainty are becoming more popular in the field of cooperative robotics. For instance, in

[1, 2] different probabilistic approaches for mapping and tracking with robotic networks are presented. [3, 4] also propose decentralized Bayesian approaches for multiple-target optimal search. Usually, these approaches are greedy, in the sense that only the next action is considered, and no planning is involved. Additionally, [5, 6] make a distinction between cooperative and coordinated information-theoretic solutions, and propose the latter to control fleets of robots. In a coordinated approach, the members of the team have no knowledge about the others' models or control actions, but they exchange some information that may influence implicitly other members' subsequent decisions. Hence, sharing fused perception information or the impact of others' control actions over a certain objective function, and acting locally, a coordinated behavior can be obtained.

Within the Bayesian approaches, Partially Observable Markov Decision Processes (POMDPs) techniques provide an elegant way to model the interaction of a network of COs with their environment. Based on prior knowledge of the sensors and actuators models and the environment dynamics, policies which indicate the objects how to act can be computed. Actually, these policies can be extracted by optimizing iteratively a certain value function over the belief space. The main problem with POMDPs as tools for planning under uncertainty is their computational complexity.

Therefore, approximate methods to obtain POMDP policies have been studied. In this sense, point-based algorithms [7–11] represent the value function by a set of vectors over the belief space (α-vectors) and restrict the optimization procedure to a bounded set of feasible beliefs. Particularly, [8] propose a point-based solver, called Perseus, where no computation is needed for all the belief points at every iteration, hence improving performance. In [12], an extension of Perseus is presented. It is called *Symbolic* Perseus and uses Algebraic Decision Diagrams (ADDs) [13] in order to optimize the operations in the original Perseus for factored POMDPs. Besides, [14] propose SARSOP, which maintains a tree-shaped set of reachable beliefs that is expanded at every iteration using the best policy so far.

Apart from the point-based approximations, some authors try to cope with the high dimensionality of the belief space by exploiting the idea that many systems often have mixed observability, i.e. although the COs state is not fully observable, some components may be. Thus, a factored model can be built to separate the fully and partially observable components of the state, leading to a lower-dimensional representation of the belief space. Then, MOMDPs (*Mixed Observability Markov Decision Processes*) are proposed in [15] for planning with large belief spaces. In order to overcome the computation problems derived from such large belief spaces, some components of the state are assumed to be fully observable (this is reasonable when there are available sensors which are accurate enough). Then, this new representation is used in conjunction with a point-based algorithm to compute approximate solutions.

This paper considers the application of decision making techniques considering uncertainty to teams of COs in tracking applications. In particular, the paper deals with target tracking by means of bearings-only sensors, like cameras,

carried by mobile robots. In this case, a single sensor does not provide information about the distance to the target, what makes tougher a proper tracking. Moreover, the motion of the target is not predictable. Thus, a team of COs can act together in order to gather enough information to achieve a more efficient performance.

A similar application with a single pursuer is addressed in [16] by means of POMDPs, but a sensor with range and bearing information is assumed. However, increasing the number of COs in the team or the set of possible locations would lead to a more complex POMDP. Hence, this paper describes a different method based on MOMDPs that can overcome this issue for certain applications. Particularly, the states (their positions) of the COs will be assumed as fully observable, focusing the problem on the target's uncertainty. Furthermore, instead of solving a large MOMDP considering all the COs, a decentralized approach where each CO acts without any knowledge about the others' actions is proposed. Then, an implicit coordination is achieved (according to the idea in [5]) by sharing a fused belief of the state among the team members. In this sense, sensors properties and desired objectives can be changed from one robot to another in order to combine different behavior within the team, and leading to a more reliable performance.

The paper is structured as follows: Sections 2 and 3 give some theoretical background about POMDPs and MOMDPs respectively; Section 4 presents MO-Symbolic Perseus, an extension of Symbolic Perseus [12] for solving MOMDPs; Section 5 details the coordinated tracking approach for multiple robots; and Sections 6 and 7 explain the experimental results and conclusions respectively.

2 POMDP Model

Formally, a POMDP is defined by the tuple $\langle S, A, Z, T, O, R, h, \gamma \rangle$. The meaning of every component is the following:

- *State space*: All the information available of the environment is encoded in the state, which is not fully observable. The system's state at time step t is defined by $s_t \in S$, where S is the finite set of all the possible states.
- *Action space*: Every CO in the environment can take an action each time step. This actions can modify the state in a stochastic manner. A is the finite set of possible actions, whereas $a_t \in A$ is the action taken at a certain time step.
- *Observation space*: Given a time step t, after executing an action, a CO can make a measurement or observation $z_t \in Z$, where Z is the finite set of all the possible observations. An observation is information about the environment that can be perceived by the agent.
- *Transition function*: When an cooperating object executes an action, the state can vary probabilistically. This probability density function is modeled by the transition function $T : S \times A \times S \longrightarrow [0, 1]$, where $T(s', a, s) = p(s_t = s' | a_{t-1} = a, s_{t-1} = s)$. It represents the probability of ending in the state s' if the CO performs the action a in the state s.

– *Observation function*: The observations gather information from the current state and are related probabilistically to this state. This probability density function is modeled by the observation function $O : Z \times A \times S \longrightarrow [0, 1]$, where $O(z, a, s') = p(z_t = z|a_{t-1} = a, s_t = s')$. It gives the probability of observing z if the action a is performed and the resulting state is s'.
– *Reward function*: A POMDP selects the best actions so that an utility function is optimized. Thus, the behavior of the system is determined by this cost function, which is modeled by a reward function. $R : S \times A \longrightarrow \Re$ is the reward function, where $R(s, a)$ is the reward obtained by executing action a in state s. Here, the reward is assumed to be bounded. Since it allows to model quite complex goals, the reward function is a very powerful tool, and hence, its right design is crucial.
– *Horizon and discount factor*: Then, the goal of a CO is to maximize the expected utility earned over some time frame. The horizon h defines this time frame by specifying the number of time steps the CO must plan for. Thus, the objective is to maximize the sum:

$$E\left[\sum_{t=0}^{h} \gamma^t r_t\right] \tag{1}$$

where r_t is the reward at time t, $E[\ ]$ is the mathematical expectation, and $\gamma \in [0, 1)$ is a discount factor, which ensures that the sum in eq. 1 is finite when $h \to \infty$. Moreover, this factor indicates how rewards should be weighted at different time steps.

Given that it is not directly observable, the actual state cannot be known by the CO. Instead, the information about the environment must be encoded in a *belief distribution*, which indicates a probably density function over the state space.

In order to calculate the current belief state $b_t(s)$ from a initial belief state $b_0(s)$, a complete trace of all the observations made and actions executed until t would be necessary. This trace is called the *history* of the system and grows as time goes by. However, due to the Markov assumption, the belief can be updated recursively using only the previous belief $b_{t-1}(s)$, the most recent action a and the most recent observation z. This belief update can be performed with a Bayesian filter and the update equation is defined by:

$$b_t(s') = \eta O(z, a, s') \sum_{s \in S} T(s', a, s) b_{t-1}(s) \tag{2}$$

where η acts as a normalizing constant such that b_t remains a probability distribution.

Once the CO has computed the belief state at a certain time, the next step is to choose which is the best action for that given belief. This action is determined by a strategy or policy $\pi(b)$, which defines the action to execute for all possible beliefs that may be encountered. The optimal policy is an attempt to map beliefs to actions so that the amount of reward earned over the time horizon

is maximized. Thus, the main objective of a POMDP algorithm is to find this policy in the form:

$$\pi(b) \longrightarrow a \tag{3}$$

where b is a belief distribution and a is the action chosen by the policy π.

The policy $\pi(b)$ is a function over the continuous set of belief states, and can be characterized by a value function $V^{\pi}(b)$, which is defined as the expected future discounted reward that the CO can gather by following π starting from belief b:

$$V^{\pi}(b) = E\left[\sum_{t=0}^{h} \gamma^t r(b_t, \pi(b_t)) | b_0 = b\right] \tag{4}$$

where $r(b_t, \pi(b_t)) = \sum_{s \in S} R(s, \pi(b_t)) b_t(s)$. Therefore, the optimal policy π^* is the one that maximizes that value function V^{π}:

$$\pi^*(b) = \arg\max_{\pi} \ V^{\pi}(b) \tag{5}$$

Note that computing an optimal policy for a POMDP may be challenging mainly for two reasons. The first one is the dimensionality of the belief space. Even for a finite set of $|S|$ states, π is defined over a $(|S| - 1)$-dimensional continuous belief space, which can be quite complex depending on the problem. The second source of complexity is the length of the history. A POMDP can be solved by searching through the space of possible histories, but the number of distinct possible action-observation histories grows exponentially with the planning horizon. An algorithm to solve POMDPs should take into account both sources of complexity in order to be efficient.

3 MOMDP Model

As mentioned before, high-dimensional belief spaces are a remarkable source of complexity when solving POMDPs. Mixed Observability Markov Decision Processes (MOMDPs) can be used in order to alleviate such a dimensionality. Many systems present *mixed observability*: although the state is not fully observable, some components may be. The idea is to model these kinds of systems with a factored POMDP where the fully and not fully observable parts of the state are separated. This model is called a MOMDP and can be combined with a point-based solver to solve traditional POMDPs [15].

Intuition says that extracting the fully observable part, the resulting POMDP that has to be solved has lower dimensionality. Therefore, the key in MOMDPs is to gain computational efficiency by solving a number of lower dimensional POMDPs instead of the original one. Thus, all operations in the point-based solver have to work on lower dimensional belief spaces, which can lead to a relevant improvement in the performance.

The mathematical model for a MOMDP is quite similar to the original POMDP and it is introduced in [15]. The MOMDP is represented as a factored POMDP where the state vector is composed of two different parts. Component x is the fully observable part of the original state s and y is another vector representing the partially observable part. Thus, the state is specified by $s = (x, y)$, and the state space is $S = X \times Y$, where X is the set with all possible values for x and Y all possible values for y.

Formally, the MOMDP is defined by the tuple $\langle X, Y, A, Z, T_x, T_y, O, R, h, \gamma \rangle$. The components are the same as in the POMDP case, but the transition function T is now decomposed into T_x and T_y. $T_x(x', a, x, y) = p(x_t = x' | a_{t-1} = a, x_{t-1} = x, y_{t-1} = y)$ gives the probability that fully observable state component has value x' if the CO takes action a in state (x, y). $T_y(y', x', a, x, y) = p(y_t = y' | x_t = x', a_{t-1} = a, x_{t-1} = x, y_{t-1} = y)$ gives the probability that the partially observable state component has value y' if the CO takes action a in state (x, y) and the fully observable state component has value x'.

Apart from the above, the other major difference between the POMDP and the MOMDP is the representation of the belief space B. Now, since x can be known at every moment, there is no need to maintain a belief over x. Therefore, x can be excluded in order to just maintain a belief $b_y(y)$, which is a probability distribution over y. Any belief $b \in B$ on the complete system state $s = (x, y)$ is then represented as (x, b_y), where $b_y \in B_y$. Furthermore, for each value x of the fully observable state component, a belief space $B_y(x) = \{(x, b_y) | b_y \in B_y\}$ is associated. Here, every $B_y(x)$ is a subspace in B, and B is a union of these subspaces $B = \bigcup_{x \in X} B_y(x)$.

Note that while B has $|X||Y|$ dimensions, each $B_y(x)$ has only $|Y|$ dimensions. Therefore, the objective of representing a high-dimensional space as a union of lower-dimensional subspaces is achieved. Moreover, when $|Y|$ is small, a remarkable computational improvement can be reached, since operations to solve the MOMDP are performed over the subspaces $B_y(x)$.

Now, since every belief is represented by (x, b_y), in [15] it is proven that the value function can be described as:

$$V(x, b_y) = \max_{\alpha \in \Gamma_y(x)} \alpha \cdot b_y \tag{6}$$

where for each x, $\Gamma_y(x)$ is a set of α-vectors defined over $B_y(x)$. Therefore, the value function is now a collection of sets of $|Y|$-dimensional vectors, and two steps are necessary in order to calculate the optimal action. First, depending on the value of the observable component x, the corresponding set of α-vectors $\Gamma_y(x)$ is selected. Then, the maximum over $\Gamma_y(x)$ is calculated. Note that all the vectors in $\Gamma_y(x)$ have only $|Y|$ dimensions instead of $|X||Y|$ like in the original POMDP. Hence, policy execution is faster for a MOMDP.

4 Symbolic Perseus with mixed observability

Once MOMDPs have been introduced, the question is how to solve them. Fortunately, with small modifications, the same point-based algorithms for POMDPs

are valid now. Since the value function consists of a set of α-vectors for every $x \in X$, the idea is to run an independent value iteration for each of them. Thus, the resulting algorithm is basically the same but dealing with each set of vectors $\Gamma_y(x)$ separately.

In [15], for instance, the point-based solver SARSOP for POMDPs is modified in order to cope with MOMDPs. Here, Symbolic Perseus [12] has been modified in order to solve MOMDPs. The resulting algorithm has been named MO-Symbolic Perseus. Recalling that Symbolic Perseus exploits the factored representation of POMDPs and the ADD structures in order to optimize the original Perseus, a combination of both, MOMDP models and Symbolic Perseus, will lead to a more efficient algorithm able to solve larger problems.

MO-Symbolic Perseus is described in algorithm 1. The structure is quite similar to the original algorithm, but some variations are needed. First, a different set of beliefs $B_y(x)$ over the component y must be sampled for every value of x. In case y is probabilistically independent of x, the same set of beliefs over y could be used for every x. All the value functions $\Gamma_y(x)$ are also initialized separately with the same values as in the original Symbolic Perseus. Note that $R_a^x(y) = R(x, y, a)$. Then, the rest of the procedure is similar to the original one but being repeated for every $x \in X$. Moreover, all the operations are made now with vectors of $|Y|$ dimensions, which simplifies some steps.

Algorithm 1 MO-Symbolic Perseus

for each $x \in X$ **do**
  Sample set of reachable beliefs $B_y(x)$
  Initialize $\Gamma_{y,0}(x) = \{R_a^x | a \in A\}$
end for
$n = 0$
repeat
  $n = n + 1$
  for each $x \in X$ **do**
    $\Gamma_{y,n}(x) = \emptyset$
    $\tilde{B} = B_y(x)$
    while $\tilde{B} \neq \emptyset$ **and** $|\Gamma_{y,n}(x)| < MAXSIZE$ **do**
      Sample $b_y \in \tilde{B}$
      $\tilde{B} = \tilde{B} \setminus \{b_y\}$
      $\alpha^*(y) = \texttt{backup}(b_y)$
      if $\Gamma_{y,n}(x) = \emptyset$ **or** $\max_{b_y \in B_y(x)}(b_y \cdot \alpha^* - V_n(x, b_y)) > 0$ **then**
        $\Gamma_{y,n}(x) = \Gamma_{y,n}(x) \cup \{\alpha^*\}$
      end if
      $\tilde{B} = \{b_y \in \tilde{B} : V_n(x, b_y) < V_{n-1}(x, b_y)\}$
    end while
  end for
until convergence
return $\Gamma_{y,n}(x)$ $\forall x$

Nonetheless, the two main steps to adapt point-based POMDP solvers to deal with MOMDPs are the belief update and the backup operation. Now, let see how these operations vary for MO-Symbolic Perseus.

In a POMDP, the belief update when an agent takes an action a and measures an observation z was:

$$b_a^z(s') = \eta p(z|a,s') \sum_{s \in S} p(s'|a,s) b(s) \tag{7}$$

where η is a normalizing constant. Besides, for a MOMDP $b_y(y) = b(s)$ when $s = (x,y)$. Hence, equation 7 can be rewritten as:

$$b_{a,y}^z(y') = \eta p(z|a,x',y') \sum_{y \in Y} p(x',y'|a,x,y) b_y(y) \tag{8}$$

$$= \eta p(z|a,x',y') \sum_{y \in Y} p(y'|x',a,x,y) p(x'|a,x,y) b_y(y) \tag{9}$$

$$= \eta O(z,a,x',y') \sum_{y \in Y} T_y(y',x',a,x,y) T_x(x',a,x,y) b_y(y) \tag{10}$$

Note that in this case, there is no need to sum over all the possible values of x and x'. Hence, equation 10 is particularized for specific values of these variables. This is due to the fact that, when updating the belief, the values of the observable states before (x) and after (x') taking an action are known.

The other major modification in MO-Symbolic Perseus respect to its original version is the backup operation. Based on all the considerations made for MOMDPs, [15] shows how the backup operation included in Perseus can be rewritten for MOMDPs:

$$\texttt{backup}(b_y) = \arg\max_{a \in A} \; b_y \cdot \alpha_a, \text{ where} \tag{11}$$

$$\begin{aligned} \alpha_a(y) = & R_a^x + \gamma \sum_{z \in Z} \sum_{x' \in X} \sum_{y' \in Y} (T_x(x',a,x,y) \\ & \times T_y(y',x',a,x,y) O(z,a,x',y') \alpha_{a,x',z}(y')) \end{aligned} \tag{12}$$

Finally, note that:

$$\alpha_{a,x',z}(y') = \arg\max_{\alpha \in \Gamma_{y,n-1}(x')} \; \alpha(y') \cdot b_{a,y}^z(y') \tag{13}$$

In this case, unlike in eq. 10, all the possible x' must be taken into account. Even though x' and z are concrete values, they cannot be known beforehand like x, so all their values must be considered and weighted by their probabilities.

5 MOMDP for Target Tracking by Mobile Cooperating Objects

An application for tracking a target by means of multiple mobile COs is considered here. In this problem there is a moving target and a team of N robots which are the pursuers. Each of these robots carries a sensor (it may be a camera) which determines whether the target is visible or not. Then, the objective is to find the target in the environment and localize it as well as possible.

The state is composed of the position of the target and the position of the pursuer robots. Since bearing-only sensors are used, the heading of every robot is also considered and included in the state. Moreover, the space is discretized into a cell grid, and a map of the scenario is assumed to be known. Thus, an occupancy grid can be obtained indicating which cells are attainable and which are obstacles. Then, the locations of the target and every pursuer are specified by cells, and just non-occupied cells are possible values. There are also four possible headings for every robot: *north*, *west*, *south* or *east*.

Every sensor provides a boolean measurement: *detected* or *non-detected*. These sensors proceed as it follows, if the target is out of its field of view, the sensor produces a *non-detected* measurement. However, when the target is within its field of view, it can be *detected* with a probability p_D.

In addition, each robot can choose at each time step among four possible actions: *stay*, *turn right*, *turn left* or *go forward*. *stay* means doing nothing; when *turning*, the robot changes its heading 90^o degrees; and when *going forward*, it moves to the cell ahead.

Finally, the key point is how to design the reward function so that the target is localized and tracked by the team of robots. Since some cooperation is desirable within the heterogeneous team, a different behavior and sensor's features are assigned to each robot. Thus, the sensor's field of view and p_D can vary from one robot to another and the reward function also depends on the specific robot: $\{R^1(x, y, a), R^2(x, y, a), \cdots, R^N(x, y, a)\}$. For all the members of the team, no cost is assigned to the action *stay*, whereas a cost of 1 is associated to the other actions.

This application is a fair example of how MOMDPs can help to reduce the belief space dimensionality. Here, even though robots and target locations are considered within the state, the one involving a greater uncertainty is the latter. Actually, for this application, the locations of the robots may be assumed to be observable (robot position could be obtained accurately enough by means of the on-board sensors and the available map), remaining as non-observable the target's position. Thus, the non-observable part of the state consists of the target location, $y = t_l$, whereas the fully observable part consists of all the robots locations and headings, $x = (r_l^1, r_h^1, \cdots, r_l^N, r_h^N)$.

The first option is to solve the above MOMDP for the whole team, but that would not be a very scalable approach regarding the number of robots. Hence, a different scheme is proposed where each robot i solves its own MOMDP without considering the other robots. That MOMDP has states $x^i = (r_l^i, r_h^i)$ and $y^i = t_l$, and reward $R^i(x, y, a)$, being possible to specify a different strategy for each

robot. Then, once the policies have been calculated, the coordination is achieved during the execution phase by sharing a fused belief state $b_{cen}(y)$ that considers information from all the robots. If $a^J = \langle a^1, \cdots, a^N \rangle$ is the joint action and $z^J = \langle z^1, \cdots, z^N \rangle$ the joint measurement, $b_{cen}(y)$ is updated in a centralized manner according to eq. 10:

$$b'_{cen}(y') = \eta p(z^J | a^J, x', y') \sum_{y \in Y} p(x', y' | a^J, x, y) b_{cen}(y) \tag{14}$$

This requires that the robots send their observations and actions to a central node that computes the fused belief. With this approach based on MOMDPs, coordination arises implicitly due to the fused belief, and there is no need to solve the original POMDP, which was far more complex. For instance, if the grid had 10x10 cells, there would be 400 possible locations for each pursuer and 100 for the target, which means a POMDP with $N \times 40,000$ -dimensional belief space. However, assuming the proposed MOMDP approach, the belief space for each of the N models becomes a union of 400 disjoint 100-dimensional subspaces. The drawback is that the policy is sub-optimal, as the robots do not reason about the other robot actions, but through proper design of the rewards and the shared information between robots it can be obtained a coordinated behavior which can be helpful for many applications.

6 Experiments

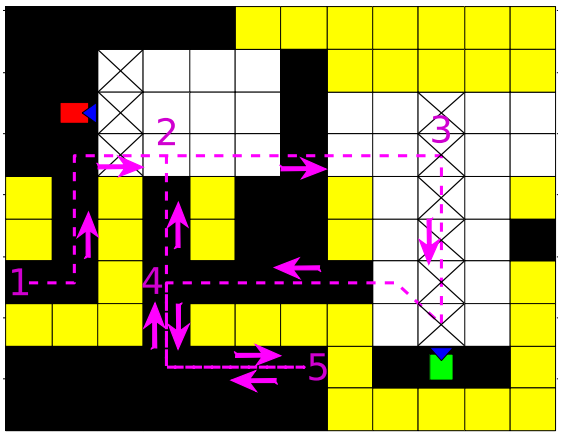

Fig. 1: Simulated environment and fields of view for each robot. If the target is in one of the cells with crosses, a high reward is obtained.

In order to test the proposed methods for target tracking, several simulations are presented here. The simulation environment has been created from the

original testbed of the Cooperating Objects Network of Excellence (CONET)[4]. This testbed considers several robots and a wireless sensor network, and allows testing with networks of static and mobile heterogeneous Cooperating Objects. As a first step, the algorithms in this paper has been tested in a simulated version of this testbed. Nonetheless, the final goal of this work is to achieve an actual implementation with the real testbed. Thus, the map of the real testbed in CONET was discretized into 2x2-meter cells and resulted in the occupancy grid of 12x10 dimensions shown in Fig. 1, where cells representing obstacles are in yellow.

For the simulations, a team with two heterogeneous robots is considered. Both carry bearing-only sensors, although their capabilities vary. The first robot (red robot) carries a more accurate sensor ($p_D = 0.9$) but its field of view is smaller, whereas the second (green robot) has a wider field of view but is less precise ($p_D = 0.8$). The fields of view for both robots are represented by white cells in Fig. 1. That is reasonable, because many vision-based detectors work more precisely when the field of view of the scene is more restricted.

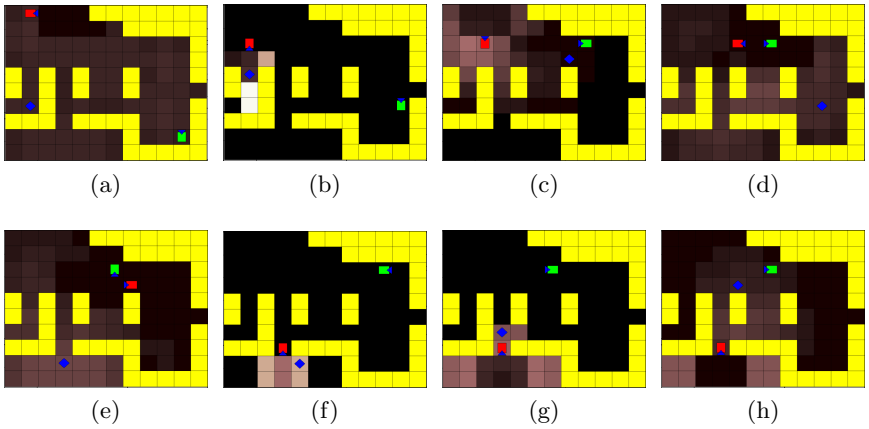

Fig. 2: Screenshots of the experiment using totally independent robots. The color scale for the cells represents the red robot's belief: the lighter, the higher the probability of being the target in the cell. The actual position of the target is represented by the blue diamond.

Therefore, the role of the robot with the wider field of view would be to survey a big area from a distant position whereas the robot with the more accurate sensor would try to get closer to confirm the target detection. This behavior can be designed through different reward functions for each robot. Hence, the red robot gets a high reward (+100) when the target is in one of the closest cells of

[4] http://www.cooperating-objects.org

its field of view, but the green robot gets a high reward (+100) when the target is in one of the central cells of its field of view. The cells getting these rewards are marked with a cross in Fig. 1, otherwise the reward is zero.

In these experiments, no uncertainty is considered for the robots' control. Thus, every time a robot takes an action, it is supposed to be executed perfectly. Nevertheless, there is some uncertainty associated to the observations and the target model. Actually, the target is assumed to move randomly. Therefore, the transition function for its position t_l indicates that, from one time step to the next, the target can move to any of its 8-connected cells with the same probability (only non-obstacle cells are considered in order to calculate that probability).

Two experiments are presented here. In the first one, each robot calculates its policy independently and run it without considering the other. In the second experiment, the approach proposed in this paper is tested. Even though the policies are calculated separately, the robots share a belief during the execution. In both experiments, the red robot starts at the top left corner and the green at the bottom right corner. Then, the target follows the fixed path depicted in Fig. 1. It starts in point 1 and follows the sequence 1-2-3-4-5-4-2, staying then within the loop 2-3-4. Moreover, in order to include some uncertainty in its behavior, at every time step, the target could (with equal probability) either stay in the same cell or follow the path. Note that all the policies were calculated with the algorithm MO-Symbolic Perseus in Section 4 by setting the $MAXSIZE$ parameter to 200 and using the same 150 sample set B_y for each $x \in X$.

Results of the first experiment are shown in Fig. 2. Since each robot looks for the target on its own, they encounter some difficulties due to the lack of range measurements. Of course, when one of them detects the target, there is no coordination and the other keeps searching around (See for example screenshots 2c, 2f and 2h). In fact, according to the belief in 2c, even though the green robot is detecting the target, the red one still thinks that it is in another area. In general, it should be noticed that the robots avoid narrow areas where their field of view could be reduced. For instance, in screenshot 2b the red robot prefers to wait for the target outside the corridor instead of going in. That is because it knows that there is no other way out.

The results of the second experiment are summarized in Fig. 3. In this case, the behavior of the individuals is similar to the first case. However, the localization of the target is now improved, since more measurements are taken and more area covered. Moreover, a certain coordination arises between the robots. For instance, in screenshot 3b the red robot detects the target and then, the green robot starts moving to the same area despite not having detected anything. Something similar happens in screenshots 3c and 3d, where the red robot loses the target but turns to the area where the other robot is detecting it. Besides, in screenshots 3f and 3g, they both lose the target and adopt a cross configuration in order to cover a wider area until they recover it. Then, in screenshot 3i the red robot recovers the target and both turn to that position. From screenshot 3i to 3l, it can be seen how the red robot tries to get close to the target whereas the green robot stays far covering a wider area.

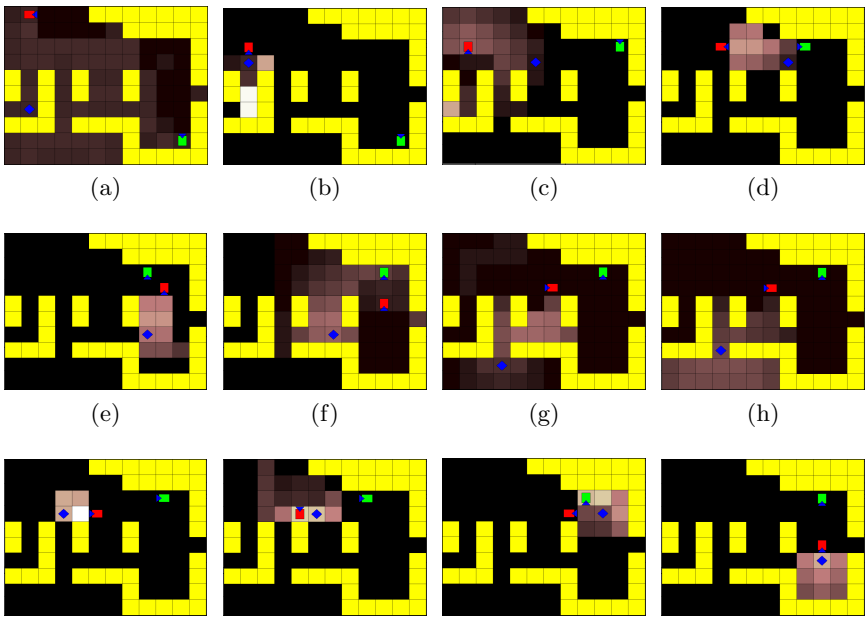

Fig. 3: Screenshots of the experiment using coordinated robots. The color scale for the cells represents the fused belief from the information of the two robots.

The same comparison, considering communication between the robots (a shared belief) and no communication, has been performed in 500 simulations considering 150 time steps (with random motion of the person and random errors in the measurements). The averaged expected reward for all the simulations has been computed. The results are shown in Table 1. It can be seen how the implicit coordination achieved through communication leads to higher rewards. The same scenario has been evaluated using SARSOP with mixed observability (presented in [15]), showing how, also in this case, the coordination leads to higher rewards.

Experiment	Reward
MO-Symbolic Perseus, belief sharing	246.13 ± 74.55
MO-Symbolic Perseus, no belief sharing	209.38 ± 70.99
SARSOP [15], belief sharing	279.64 ± 104.46
SARSOP [15], no belief sharing	197.76 ± 69.34

Table 1: Experiment with 2 robots. Average reward obtained considering communication (belief sharing), and without communication.

7 Conclusions

The paper has presented how POMDPs can be used for Cooperating Objects planning under uncertainty. However, the intractability of some models is still a drawback for large belief spaces. This work has highlighted the important role that MOMDPs can play in order to cope with problems in real applications. Thus, an example of how mixed observability can alleviate the complexity of the POMDPs has been applied to target tracking.

Moreover, another interesting idea shown in the paper is how solving the MOMDPs in a decentralized manner and sharing information can lead to implicit coordination of a team of COs, without having to solve the full POMDP model. In this sense, results with non-coordinated and coordinated robots were compared and a better performance was achieved with the latter, even though intentional cooperation was not considered. Furthermore, the combination of different policies for the heterogeneous robots was also exploited. Since each CO does not reason about the others' actions, this approach is rather scalable.

Even though the MOMDPs are solved in a decentralized way, during the execution phase the belief is fused centrally. Therefore, a future development would be to decentralize also the fusion of the beliefs so that the whole system is totally scalable. Then, the future idea of this work is to implement a totally decentralized scheme that can be tested in a real platform.

Furthermore, although one particular application has been presented, due to their generality and adaptability the same techniques can be applied to other scenarios. Currently, we are working on the activation problem (that is, policies

to decide optimally which sensors should be activated in a sensor network for tracking applications, taking into account energy limitations) and on the cooperation between static and mobile COs for cooperative navigation when the mobile COs lack of accurate sensors for localization.

8 Acknowledgments

This work is partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

References

1. Stroupe, A., Balch, T.: Value-based action selection for observation with robot teams using probabilistic techniques. Robotics and Autonomous Systems **50** (February 2005) 85–97
2. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration. Signal Processing Magazine, IEEE **19**(2) (March 2002) 61–72
3. Wong, E.M., Bourgault, F., Furukawa, T.: Multi-vehicle bayesian search for multiple lost targets. In: Proceedings of the International Conference on Robotics and Automation. (2005) 3169–3174
4. Bourgault, F., Furukawa, T., Durrant-Whyte, H.: Decentralized bayesian negotiation for cooperative search. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2004) 2681–2686
5. Grocholsky, B., Makarenko, A., Kaupp, T., Durrant-Whyte, H.F.: Scalable Control of Decentralised Sensor Platforms. In: Lecture notes in Computer Science. Volume 2634. Springer (2003)
6. Mathews, G., Durrant-Whyte, H., Prokopenko, M.: Decentralized Decision Making for Multiagent Systems. In: Advances in Applied Self-Organizing Systems. Springer-Verlag (2008)
7. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. (2003) 1025–1032
8. Spaan, M.T.J., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research **24** (2005) 195–220
9. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence. (2004) 520–527
10. Smith, T., Simmons, R.: Point-based POMDP algorithms: improved analysis and implementation. In: Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence. (2005) 542–547
11. Hauskrecht, M.: Value function approximations for partially observable Markov decision processes. Journal of Artificial Intelligence Research **13** (2000) 33–95
12. Poupart, P.: Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes. PhD thesis, University of Toronto (2005)
13. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: SPUDD: Stochastic planning using decision diagrams. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence. (1999)

14. Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science and Systems. (2008)
15. Ong, S.C., Png, S.W., Hsu, D., Lee, W.S.: POMDPs for Robotic Tasks with Mixed Observability. In: Robotics: Science and Systems Conference. (2009)
16. Hsu, D., Lee, W., Rong, N.: A point-based POMDP planner for target tracking. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2008) 2644–2650